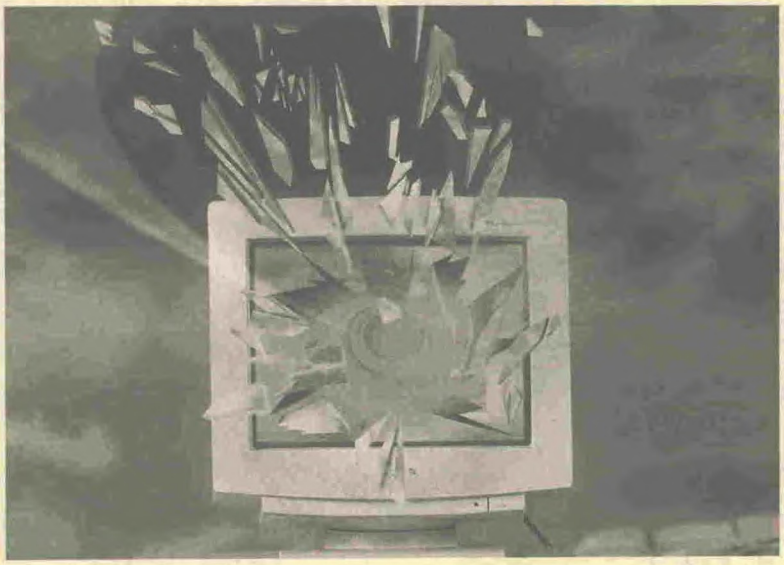
สุทธิศักดิ์ พงศ์ธนาพาณิช

เปิดโลกมัลติมีเดียด้วย VB/Win

ตอน การเขียนโปรแกรมมัลติมีเดียสำหรับวินโดวส์ 95 ด้วย VB/Win



เทคนิคสำหรับการเขียนโปรแกรม ด้วย Visual BASIC 4.0

สำหรับบทความเปิดโลกมัลติมีเดียด้วย VB/Win นี้ ก็มีอายุล่วงเลยมากว่า 1 ปีแล้ว โดยมีวัตถุประสงค์ที่จะแนะนำเทคนิคการนำฟังก์ชันวินโดว์ API มาประยุกต์ใช้งานกับ VB/Win เพื่อสร้างแอพพลิเคชันด้านมัลติมีเดีย ซึ่งสามารถทำงานได้เฉพาะวินโดวส์ 3.x เท่านั้น แต่ในปัจจุบันผู้เขียนเห็นว่าน่าจะเป็นการเหมาะสมที่เราจะหันมาให้ความสนใจกับการเขียนแอพพลิเคชันด้านมัลติมีเดียสำหรับวินโดวส์ 32 บิตกันบ้าง เพราะแนวโน้มในอนาคตนี้น่าจะเป็นยุคต์ของระบบปฏิบัติการ 32 บิตอย่างสมบูรณ์ ในปัจจุบันไมโครซอฟท์ได้เสนอวินโดวส์ 95 ซึ่งเป็นระบบปฏิบัติการ 32 บิต (แต่ยังคงมีส่วนของโค้ดที่สนับสนุนแอพพลิเคชันระดับ 16 บิตอยู่เป็นจำนวนมาก) เพื่อที่จะแทนที่วินโดวส์ 3.x แต่เนื่องจากวินโดวส์ 95 เป็นระบบปฏิบัติการที่ยังคงมีปัญหาด้านความน่าเชื่อถือและความเสถียรอยู่มากพอสมควร (เมื่อเปรียบเทียบกับวินโดวส์ NT หรือ OS/2) ดังนั้นในช่วงปีที่ผ่านมาซึ่งเป็นช่วงที่วินโดวส์ 95 ได้ออกมาใหม่ ๆ ผู้เขียนจึงไม่หันไปเขียนบทความเพื่อสนับสนุนการเขียนโปรแกรมสำหรับวินโดวส์ 95 ทันที เพราะผู้เขียนมีความคิดเป็นการส่วนตัวว่าวินโดวส์ 3.x น่าจะยังคงมีส่วนแบ่งในตลาดมากกว่าวินโดวส์ 95 อยู่เป็นจำนวนมาก โดยเฉพาะในบ้านเรา

ถึงแม้ว่าจะเป็นช่วงเวลาในขณะนี้ก็ตามวินโดวส์ 3.x ก็ยังคงได้รับความนิยมในหมู่ผู้ใช้แต่เดิมอยู่มากพอสมควร ยกเว้นก็แต่ผู้ใช้ที่ซื้อเครื่องคอมพิวเตอร์ในช่วง 10 เดือนที่ผ่านมาเท่านั้น ที่ไมโครซอฟท์ได้เสนอกลยุทธ์ที่จะสนับสนุนวินโดวส์ 95 แทนวินโดวส์ 3.x โดยให้ผู้จำหน่ายเครื่องได้ติดตั้งวินโดวส์ 95 ไปพร้อมกับเครื่องใหม่ในราคาที่ถูก จึงทำให้ระบบปฏิบัติการที่ติดมากับเครื่องทุก ๆ เครื่องในปัจจุบันจึงเป็นวินโดวส์ 95

เมื่อแนวโน้มของตลาดในอนาคตน่าจะเป็นยุคของระบบปฏิบัติการ 32 บิต ผู้เขียนจึงได้มีแนวความคิดที่จะเปลี่ยนแนวการเสนอบทความนี้ เพื่อเป็นการแนะนำการใช้ VB/Win สร้างแอพพลิเคชันสำหรับวินโดวส์ 95 โดยจะพยายามแนะนำฟังก์ชันวินโดว์ API ที่มีความคอมแพตทิเบิลสำหรับวินโดวส์ 95 และวินโดวส์ NT เพราะผู้เขียนมีความเชื่อเป็นการส่วนตัวว่าในอนาคตแก่นแท้ของการจัดการของระบบปฏิบัติการวินโดวส์ (kernel) น่าจะยึดถือวินโดวส์ NT เป็นแม่แบบ เนื่องจากเป็นระบบปฏิบัติการที่มีความน่าเชื่อถือและความเสถียรสูงมากนั่นเอง สำหรับ VB/Win 4.0 ก็เป็นเวอร์ชันแรกที่สนับสนุนการทำงานภายใต้วินโดวส์ 16 บิต และวินโดวส์ 32 บิต และเป็นเวอร์ชันที่ได้รับการปรับปรุงให้มีความสามารถและความเสถียรดีขึ้นกว่าเดิมเป็นอย่างมาก

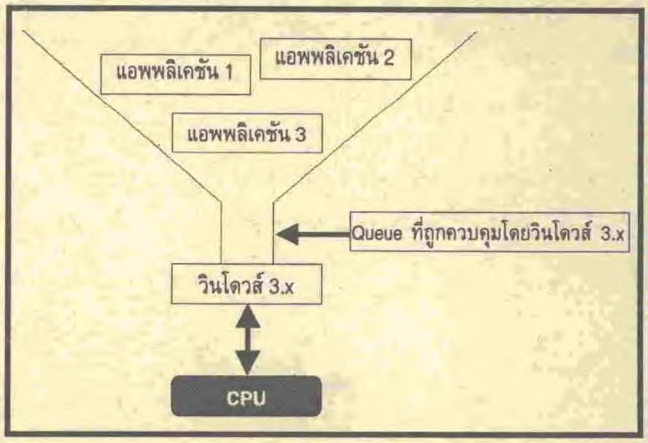
VB/Win เป็นภาษาที่ง่ายต่อการใช้งาน และในการสร้างแอพพลิเคชันด้านมัลติมีเดีย ผู้อ่านก็สามารถใช้ VB/Win 4.0 ในการสร้างงานเหล่านี้ได้โดยไม่ยากเลย และนอกจากนี้ฟังก์ชันวินโดว์ API ระดับ 32 บิต โดยส่วนใหญ่ก็ยังคอมแพตทิเบิลกับ VB/Win และนอกจากนี้ฟังก์ชันวินโดวส์ API 32 บิต บางส่วนก็ได้ถูกดึงมาเป็นส่วนหนึ่งของ VB/Win 4.0 โดยตรง ดังนั้นจึงทำให้ผู้อ่านสามารถลดเวลาในการสร้างแอพพลิเคชันได้มากทีเดียว และที่สำคัญคำสั่งของ VB/Win 4.0 ล้วนแต่ได้รับการปรับแต่ง (optimization) มาเป็นอย่างดีอยู่แล้ว จึงทำให้คำสั่งที่ได้สามารถทำงานได้รวดเร็วกว่าการนำฟังก์ชันวินโดว์ API 32 บิต มาเขียนเป็นคำสั่งเอาเอง โดยที่ผู้อ่านสามารถสังเกตได้จากรายงานจำนวนผู้ใช้ VB/Win จากวารสารต่างประเทศ เพราะในปัจจุบันได้มีโปรแกรมเมอร์บางกลุ่มได้หันมาใช้ VB/Win แทน C++ หรือ Visual C++ ทั้งนี้เนื่องจากง่ายต่อการใช้งาน และใช้เวลาในการพัฒนาแอพพลิเคชันน้อย และที่สำคัญเมื่อโค้ดได้รับการออพติไมซ์ ตามอัลกอริทึมที่เหมาะสม ก็จะทำให้ความแตกต่างด้านความเร็วของการทำงานลดลงไปได้เป็นอย่างมาก โดยเฉพาะเมื่อทำงานภายใต้ CPU ระดับเพนเทียมก็จะไม่สามารถสังเกตความแตกต่างด้านความเร็วในการทำงานได้เลย

สำหรับผู้อ่านที่ยังคงเป็นมือใหม่สำหรับระบบปฏิบัติการวินโดวส์ 95 หรือ VB/Win 4.0 ก็ขอแนะนำให้ศึกษาเพิ่มเติมได้จากหนังสือ Visual Basic 4.0 Professional แต่งโดยผู้เขียนเอง และจัดพิมพ์โดย บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน) และจากบทความ Advanced BASIC Programming Technique ในตอนการเขียนโปรแกรมสำหรับวินโดวส์ 95 ด้วย VB/Win 4.0-32 บิต ซึ่งได้จัดพิมพ์ลงในวารสารไมโครฯ ฉบับที่ 143 ประจำเดือนมิถุนายน ซึ่งจะแนะนำพื้นฐานการแปลง (จากโค้ดของ C++) และประยุกต์ฟังก์ชันวินโดว์ API สำหรับ VB/Win อย่างละเอียดครับ

มัลติทาสกิงและมัลติทริดดิงส์

เนื่องจากแอพพลิเคชันด้านมัลติมีเดียส่วนใหญ่จะใช้พลังการประมวลผลของ CPU เป็นอย่างมาก ดังนั้นก่อนที่เราจะกล่าวถึงการเขียนโปรแกรมมัลติมีเดียด้วย VB/Win ก็จะขออธิบายถึงแนวคิดของการแบ่งสรรเวลาการทำงานของแอพพลิเคชันภายใต้วินโดวส์ที่เรียกกันว่า มัลติทาสกิงและมัลติทริดดิงส์ เพื่อให้ผู้อ่านได้เข้าใจถึงพื้นฐานด้านการจัดการแอพพลิเคชันของวินโดวส์ ซึ่งจะช่วยเสริมให้ผู้อ่านสามารถนำไปประยุกต์ออกแบบแอพพลิเคชันได้อย่างถูกต้องและทำงานได้อย่างรวดเร็ว

ครั้งแรกที่ไมโครซอฟท์ได้เข็นวินโดวส์ 3.x ออกสู่ตลาด วินโดวส์ 3.x ก็ได้รับการกล่าวขานถึงความสามารถด้านการทำงานแบบมัลติทาสกิง (multitasking) ผู้อ่านบางท่านอาจจะไม่ทราบว่ามัลติทาสกิงคืออะไร หรือมีความเกี่ยวข้องอย่างไรกับ VB/Win ก็ขอให้อ่านบทความส่วนนี้ต่อไปนะครับ คำว่ามัลติทาสกิงสำหรับวินโดวส์ 3.x นั้นในความเป็นจริงก็หมายถึง non-preemtive multitasking นั่นเอง ซึ่งเป็นระบบปฏิบัติการที่อนุญาตให้หลาย ๆ แอพพลิเคชันสามารถทำงานร่วมกันได้ในเวลาเดียวกัน โดยที่วินโดวส์จะทำหน้าที่บริหารและจัดสรรทรัพยากรของระบบและ CPU ให้กับแต่ละแอพพลิเคชันตามลำดับความต้องการและความสำคัญ ซึ่งวินโดวส์จะทราบสิ่งเหล่านี้ได้จาก message ที่แต่ละแอพพลิเคชันติดต่อกับวินโดวส์ในขณะทำงาน แต่มัลติทาสกิงสำหรับวินโดวส์ 3.x นั้นไม่ใช้ระบบที่การควบคุมทุก ๆ อย่างจะถูกกระทำโดยวินโดวส์โดยลำพัง เพราะวินโดวส์จะสามารถให้บริการหรือสวิทช์การทำงานจากแอพพลิเคชันหนึ่งไปยังแอพพลิเคชันอื่น ๆ ได้ก็ต่อเมื่อแอพพลิเคชันที่ควบคุมการใช้งาน CPU ในขณะนั้น ได้มีการแจ้งให้วินโดวส์ทราบถึงจุดสิ้นสุดการทำงานของงาน (task) แล้วเท่านั้น นั่นก็หมายความว่าถ้าหากงานของแอพพลิเคชันยังไม่สิ้นสุดการทำงานหรือเกิดค้าง (hang) แอพพลิเคชันอื่น ๆ ก็จะไม่สามารถทำงานได้เลย ซึ่งผู้อ่านสามารถสังเกตได้จากในระหว่างที่โปรแกรมไมโครซอฟต์ Access กำลังประมวลผล SQL อยู่นั้น ผู้อ่านจะไม่สามารถสวิตช์การทำงานไปยังแอพพลิเคชันอื่น ๆ ได้เลย นอกจากจะรอให้การประมวลผล SQL สิ้นสุดแล้วเท่านั้น ดังนั้นถ้าหากแอพพลิเคชันที่กำลังใช้งาน CPU เกิดค้าง (hang) เนื่องจากเกิดข้อผิดพลาดเช่น GPF ก็อาจจะส่งผลให้แอพพลิเคชันอื่น ๆ ไม่สามารถทำงานตามไปด้วย ซึ่งในกรณีที่เลวร้ายก็สามารถทำให้วินโดวส์ค้างได้ทันที เพื่อให้ง่ายต่อความเข้าใจขอให้ผู้อ่านสังเกตการจำลองแนวคิดของมัลติทาสกิงจากรูปที่ 1



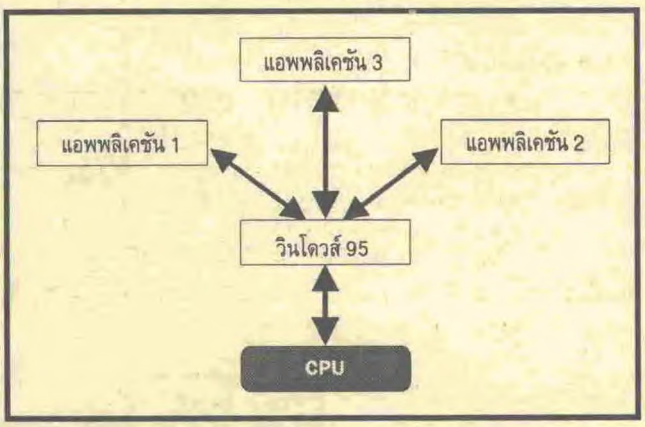
รูปที่ 1 แสดงแนวคิดของมัลติทาสกิงส์สำหรับวินโดวส์ 3.x

สำหรับวินโดวส์ 95 หรือ NT นั้นไมโครซอฟท์ได้นำแนวคิดใหม่เกี่ยวกับมัลติทาสกิงมาใช้ โดยใช้ชื่อเรียกว่า preemtive multitasking หรือมัลติทริดดิงส์ โดยที่ระบบจัดการด้านการจัดสรรเวลาในการทำงานของแอพพลิเคชันนั้นจะถูกควบคุมโดยวินโดวส์อย่างสมบูรณ์ โดยที่วินโดวส์จะทำหน้าที่จัดตารางการทำงานของแอพพลิเคชัน (slice time schedule) ให้โดยอัตโนมัติ ดังนั้นถึงแม้ว่าในขณะนั้นยังคงมีแอพพลิ-

เคชันหนึ่งทำงานอยู่ก็ตาม วินโดวส์จะไม่รอจนกว่าแอพพลิเคชันนั้นจะสิ้นสุดการทำงานของงาน (task) หนึ่ง ๆ ของแอพพลิเคชัน แต่วินโดวส์จะจัดสรร CPU ให้กับแอพพลิเคชันอื่น ๆ ตามลำดับที่ปรากฏในตารางของวินโดวส์ได้ใช้งานต่อไป ด้วยเหตุนี้จึงไม่มีแอพพลิเคชันใดเลยที่ทำงานภายใต้วินโดวส์ 95 ที่จะสามารถใช้งาน CPU ได้เพียงแอพพลิเคชันเดียวในขณะที่กำลังประมวลผลงานหนึ่ง ๆ ของตนเอง

เนื่องจากวินโดวส์ 95 ไม่อนุญาตให้แอพพลิเคชันอ้างสิทธิ์การใช้งาน CPU เกินกว่าเวลาที่วินโดวส์กำหนดไว้ให้ นั่นก็หมายความว่าวินโดวส์ 95 ไม่ได้มองโค้ดที่กำลังทำงานในรูปแบบของงาน (task) อีกต่อไป แต่มันจะมองในลักษณะของโค้ดที่กำลังปฏิบัติการเท่านั้น ดังนั้นไมโครซอฟท์จึงได้เรียกส่วนของโค้ดที่กำลังปฏิบัติการของแอพพลิเคชันว่าทริด (thread) ดังนั้นทริดจึงสามารถหมายถึง ตัวของแอพพลิเคชันหนึ่ง ๆ หรือส่วนของโค้ดที่กำลังปฏิบัติการของแอพพลิเคชันหนึ่ง ๆ ก็ได้ ด้วยเหตุนี้ผู้อ่านจะเห็นว่าตามแนวคิดของมัลติทริดดิงส์นั้น วินโดวส์จะสามารถปฏิบัติแอพพลิเคชันได้รวดเร็วกว่า และสามารถควบคุมการใช้งาน CPU ได้อย่างมีประสิทธิภาพกว่ามัลติทาสกิง ดังนั้นในการเขียนแอพพลิเคชันด้วย VB/Win ถ้าหากผู้อ่านออกแบบแอพพลิเคชันให้เหมาะสมและสามารถใช้ประโยชน์จากมัลติทริดดิงส์ของวินโดวส์ 95 ได้แล้วละก็ แอพพลิเคชันก็จะสามารถทำงานได้รวดเร็วขึ้นนั่นเอง

แต่เนื่องจาก VB/Win ไม่สนับสนุนการสร้างแอพพลิเคชันที่ทำงานแบบมัลติทริดดิงส์อย่างสมบูรณ์ ดังนั้นแอพพลิเคชันที่เขียนด้วย VB/Win จึงควรที่จะแบ่งแยกหลาย ๆ แอพพลิเคชันตามลักษณะของงานที่ต้องการการทำงานพร้อม ๆ กัน หรืออาจจะออกแบบให้มีการทำงานแบบ out-of-process OLE Automation ก็ได้ ซึ่งวินโดวส์ 95 ก็จะจัดการด้านการใช้งาน CPU ตามแนวคิดของมัลติทริดดิงส์โดยอัตโนมัติ (ดูรูปที่ 2)

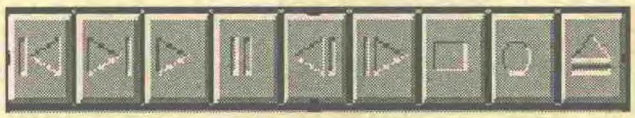


รูปที่ 1 แสดงแนวคิดของมัลติทริดดิงส์สำหรับวินโดวส์ 95

VB/Win กับมัลติมีเดียระดับ 32 บิต

โดยตัวของ VB/Win จะไม่มีคำสั่งใด ๆ เลยที่สนับสนุนการทำงานด้านมัลติมีเดีย แต่อย่างไรก็ตาม VB/Win ได้เตรียม custom control ที่ช่วยจัดการงานด้านมัลติมีเดียได้พอสมควรที่ชื่อ MCI32.OCX ซึ่งผู้อ่านสามารถใช้คอนโทรล MCI ดังกล่าวในการเล่นกลับหรืออัดเพลงในรูปแบบของไฟล์ .wav หรือ .mid ก็ได้ และนอกจากนี้ยังสามารถเล่นกลับไฟล์ .avi และ CD-Audio ได้อีกด้วย และที่สำคัญคอนโทรล MCI32.OCX ยังได้รับการปรับปรุงให้สามารถมีการโต้ตอบระหว่าง VB/Win และ mmsystem ของวินโดวส์ 95 (callback หรือ notification) ได้อีกด้วย ซึ่งจะช่วยให้เราสามารถควบคุมหรือทราบสถานะการทำงานของระบบ mmsystem ของวินโดวส์ 95 ได้ทุกขณะการทำงาน ดังนั้นในบทความตอนนี้ผู้เขียนจึงขออธิบายถึงคอนโทรล MCI32.OCX ของ VB/Win อย่างละเอียด เพื่อให้ผู้อ่านทดลองนำไปใช้งาน ให้เกิดความคุ้นเคยกับกับระบบมัลติมีเดียของวินโดวส์ 95 มากขึ้น

คอนโทรล MCI32.OCX ของ VB/Win จะปรากฏในรูปของไอคอนดังนี้ โดยเมื่อผู้อ่านนำคอนโทรลนี้ใส่ลงในฟอร์มก็จะปรากฏชุดของปุ่มสำหรับควบคุมมัลติมีเดียดังในรูปที่ 3



รูปที่ 3 แสดงชุดปุ่มสำหรับควบคุมมัลติมีเดียของคอนโทรล MCI

และเนื่องจากคอนโทรล MCI นี้ ได้รับการพัฒนาให้อิงกับฮาร์ดแวร์และซอฟต์เฉพาะสำหรับบางผู้ผลิตเท่านั้น ดังเช่น MPC, CD-ROM Upgrade Kit หรือ Audio Card Upgrade Kit เป็นต้น ดังนั้นถ้าหากระบบมัลติมีเดียของเครื่องของผู้อ่านไม่สามารถใช้งานร่วมกับคอนโทรล MCI ได้ก็ขอให้ตรวจสอบชนิดของฮาร์ดแวร์ด้านมัลติมีเดียอีกครั้ง สำหรับคุณสมบัติของคอนโทรล MCI มีดังต่อไปนี้

คุณสมบัติ AutoEnabled

รายงานหรือกำหนดความสามารถในการ enabled หรือ disable แต่ละปุ่มของคอนโทรล MCI

```
[form.]MMControl.AutoEnabled [= {True/False}]
```

True หมายถึง ปุ่มต่าง ๆ สามารถถูกกำหนดให้มีสถานะ enabled หรือ disable โดยอัตโนมัติ

False หมายถึง ปุ่มต่าง ๆ ไม่สามารถถูกกำหนดให้มีสถานะ enabled หรือ disable โดยอัตโนมัติ

คุณสมบัติ *Button*Enabled

รายงานหรือกำหนดสถานะการ enabled หรือ disable ของปุ่มของคอนโทรล MCI

```
[form.]MMControl.ButtonEnabled [= {True/False}]
```

True หมายถึง ปุ่มที่ถูกกำหนดมีสถานะ enabled

False หมายถึง ปุ่มที่ถูกกำหนดมีสถานะ disabled

สำหรับรายชื่อของแต่ละปุ่มที่สนับสนุนมีดังนี้ Back, Eject, Next, Pause, Play, Prev, Record, Step และ Stop ทั้งนี้ก็ขึ้นอยู่กับชนิดของมัลติมีเดียที่คอนโทรล MCI กำลังติดต่อด้วยเช่นกัน

คุณสมบัติ *Button*Visible

รายงานหรือกำหนดสถานะการแสดงผลของปุ่มของคอนโทรล MCI

```
[form.]MMControl.ButtonVisible [= {True/False}]
```

True หมายถึง ปุ่มที่ถูกกำหนดมีสถานะแสดงผล

False หมายถึง ปุ่มที่ถูกกำหนดมีสถานะไม่แสดงผล

สำหรับรายชื่อของแต่ละปุ่มที่สนับสนุนมีดังนี้ Back, Eject, Next, Pause, Play, Prev, Record, Step และ Stop ทั้งนี้ก็ขึ้นอยู่กับชนิดของมัลติมีเดียที่คอนโทรล MCI กำลังติดต่อด้วยเช่นกัน

คุณสมบัติ CanEject

รายงานสถานะสนับสนุนการใช้งานของปุ่ม Eject ของคอนโทรล MCI

```
[form.]MMControl.CanEject
```

True หมายถึง สนับสนุนการใช้งานของปุ่ม Eject

False หมายถึง ไม่สนับสนุนการใช้งานของปุ่ม Eject

คุณสมบัติ CanPlay

รายงานสถานะสนับสนุนการใช้งานของปุ่ม Play ของคอนโทรล MCI

```
[form.]MMControl.CanPlay
```

True หมายถึง สนับสนุนการใช้งานของปุ่ม Play

False หมายถึง ไม่สนับสนุนการใช้งานของปุ่ม Play

คุณสมบัติ CanRecord

รายงานสถานะสนับสนุนการใช้งานของปุ่ม Record ของคอนโทรล MCI

```
[form.]MMControl.CanRecord
```

True หมายถึง สนับสนุนการใช้งานของปุ่ม Record

False หมายถึง ไม่สนับสนุนการใช้งานของปุ่ม Record

คุณสมบัติ CanStep

รายงานสถานะสนับสนุนการใช้งานของปุ่ม Step ของคอนโทรล MCI

```
[form.]MMControl.CanStep
```

True หมายถึง สนับสนุนการใช้งานของปุ่ม Step

False หมายถึง ไม่สนับสนุนการใช้งานของปุ่ม Step

เฉพาะดีไวซ์มัลติมีเดีย MMMovie, Overlay และ VCR เท่านั้นที่สนับสนุนการเลื่อนไปมาระหว่างเฟรม หรือที่เรียกว่า step positioning

คุณสมบัติ Command

กำหนดคำสั่ง MCI สำหรับสั่งงานให้กับคอนโทรล MCI

```
[form.]MMControl.Command [= cmdstring$]
```

cmdstring$ หมายถึงคำสั่ง MCI ที่แต่ละชนิดของไดเวอร์สนับสนุน ซึ่งคำสั่งทั้งหมดที่สนับสนุนมีรายละเอียดดังต่อไปนี้

Open เปิดดีไวซ์ที่กำหนด

Close ปิดดีไวซ์ที่กำหนด

Play เล่นกลับดีไวซ์ที่ถูกเปิดด้วยคำสั่ง Open

Pause หยุดการเล่นกลับชั่วคราว

Stop หยุดการเล่นกลับถาวร

Back ถอยกลับไปยังเฟรมที่ผ่านมา

Step เดินหน้าไปยังเฟรมที่ถัดมา

Prev เลื่อนไปยังจุดเริ่มต้นของแทรก (track) ปัจจุบัน

Next เลื่อนไปยังจุดเริ่มต้นของแทรก (track) ถัดไป

Seek เลื่อนไปยังตำแหน่งที่กำหนด

Record บันทึกข้อมูลมัลติมีเดีย ลงบนดิสก์

Eject เปิดประตูถาดเครื่องอ่าน CD

Sound เล่นเสียง

Save จัดเก็บข้อมูลมัลติมีเดียลงบนดิสก์

คุณสมบัติ DeviceID

กำหนดหมายเลขประจำดีไวซ์ที่ถูกเปิดด้วยคำสั่ง Open ในปัจจุบัน

```
[form.]MMControl.DeviceID [= id%]
```

id% หมายถึง หมายเลข ID ประจำดีไวซ์ที่ถูกเปิด ซึ่งจะถูกกำหนดโดยวินโดวส์และรายงานให้ทราบเมื่อ ผู้อ่านเปิดดีไวซ์ด้วยคำสั่ง Open

คุณสมบัติ DeviceType

กำหนดประเภทของดีไวซ์ที่ถูกเปิดด้วยคำสั่ง Open ในปัจจุบัน

```
[form.]MMControl.DeviceType [= device%]
```

device% หมายถึง ประเภทของดีไวซ์ที่ถูกเปิด ซึ่งจะถูกกำหนดโดยวินโดวส์และรายงานให้ทราบเมื่อผู้อ่านเปิดดีไวซ์ด้วยคำสั่ง Open ซึ่งสามารถมีได้ดังต่อไปนี้ AVIVideo, CDAudio, DAT, DigitalVideo,

MMMovie, Overlay, Scanner, Sequencer, VCR, Videodisc หรือ WaveAudio

คุณสมบัติ Enabled

กำหนดหรือรายงานสถานะ enabled หรือ disabled ของคอนโทรล MCI

```
[form.]MMControl.Enabled [= {True/False}]
```

True หมายถึง คอนโทรลมีสถานะ enabled แต่ปุ่มใดจะมีสถานะ enabled ก็ขึ้นกับคุณสมบัติ *Button*Enabled หรือ AutoEnable

False หมายถึง ทุกปุ่มมีสถานะ disabled

คุณสมบัติ Error

รายงานรหัสข้อผิดพลาดของคอนโทรล MCI ในขณะทำงาน

```
[form.]MMControl.Error
```

ถ้าหากมีค่าเท่ากับ 0 แสดงว่าไม่มีข้อผิดพลาดเกิดขึ้นในขณะทำงาน

คุณสมบัติ ErrorMessage

รายงานข้อความแสดงข้อผิดพลาดของคอนโทรล MCI ในขณะทำงาน

```
[form.]MMControl.ErrorMessage
```

คุณสมบัติ FileName

รายงานหรือกำหนดชื่อไฟล์ ที่ถูกเปิดหรือจัดเก็บด้วยคำสั่ง Open หรือ Save ตามลำดับ

```
[form.]MMControl.FileName [= string$]
```

คุณสมบัติ Frames

รายงานหรือกำหนดหมายเลขเฟรมที่ต้องเลื่อนไปด้วยคำสั่ง Step หรือ Back

```
[form.]MMControl.Frames [= frame&]
```

คุณสมบัติ From, To

รายงานหรือกำหนดตำแหน่งเริ่มต้น (From) และตำแหน่งสุดท้าย (To) ตามลำดับ สำหรับการเล่นหรือบันทึก ซึ่งหน่วยของตำแหน่งเริ่มต้นจะเป็นไปตามที่กำหนดโดยคุณสมบัติ TimeFormat

```
[form.]MMControl.From [= location&]
[form.]MMControl.To [= location&]
```

คุณสมบัติ hWndDisplay

รายงานหมายเลข handle สำหรับหน้าต่างแสดงผลดีไวซ์ MMMovie หรือ Overlay

```
[form.]MMControl.hWndDisplay
```

คุณสมบัติ Length

รายงานความยาวของดีไวซ์ที่ถูกเปิด ในหน่วยของเวลาที่ถูกกำหนดโดยคุณสมบัติ TimeFormat

```
[form.]MMControl.Length
```

คุณสมบัติ Mode

รายงานโหมดปัจจุบันของดีไวซ์ที่ถูกเปิด ผู้อ่านสามารถอ่านคุณสมบัตินี้ได้เฉพาะในขณะรันแอพพลิเคชันเท่านั้น

```
[form.]MMControl.Mode
```

โหมดที่คอนโทรล MCI สนับสนุนมีดังต่อไปนี้

```
524 (mciModeNotOpen)    Open
525 (mciModeStop) Stop
526 (mciModePlay) Play
527 (mciModeRecord)      Record
528 (mciModeSeek) Seek
529 (mciModePause)       Pause
530 (mciModeReady)       Ready
```

คุณสมบัติ Notify

รายงานหรือกำหนดให้คอนโทรล MCI สร้างเหตุการณ์ callback (Done) ในขณะที่คำสั่งถัดไปทำงานเรียบร้อยแล้ว

```
[form.]MMControl.Notify [= {True/False}]
```

คุณสมบัติ NotifyMessage

รายงานสตริงรายละเอียดของรหัสการแจ้งกลับ (Notify code) สำหรับเหตุการณ์ Done

```
[form.]MMControl.NotifyMessage
```

คุณสมบัติ NotifyValue

รายงานค่าตัวเลขที่แสดงถึงผลลัพธ์ของคำสั่งสุดท้ายที่มีการร้องขอให้มีการแจ้งกลับ (Notification)

```
[form.]MMControl.NotifyValue
```

ซึ่งค่าตัวเลขที่แสดงถึงผลลัพธ์ของการแจ้งกลับ มีดังต่อไปนี้

1 (mciNotifySuccessful) คำสั่งทำงานเสร็จเรียบร้อย

2 (mciNotifySuperseded) คำสั่งถูกแทนที่หรือควบคุมโดยคำสั่งอื่น ๆ

3 (mciNotifyAborted) คำสั่งถูกยกเลิกโดยผู้ใช้

8 (mciNotifyFailure) คำสั่งทำงานไม่สำเร็จ

คุณสมบัติ Orientation

รายงานหรือกำหนดทิศทางการเรียงปุ่มคำสั่ง MCI

```
[form.]MMControl.Orientation [= orientation%]
```

0 (mciOrientHorz) เรียงปุ่มคำสั่งในแนวนอน

1 (mciOrientVert) เรียงปุ่มคำสั่งในแนวตั้ง

คุณสมบัติ Position

รายงานตำแหน่งปัจจุบันที่กำลังทำงานของดีไวซ์ที่ถูกเปิด โดยที่หน่วยของตำแหน่งปัจจุบันจะเป็นไปตามที่ถูกกำหนดโดยคุณสมบัติ TimeFormat

```
[form.]MMControl.Position
```

คุณสมบัติ RecordMode

รายงานหรือกำหนดโหมดการบันทึกสำหรับดีไวซ์ที่กำหนด

```
[form.]MMControl.RecordMode [= mode%]
```

0 (mciRecordInsert) บันทึกข้อมูลแทรกต่อจากข้อมูลเดิมของดีไวซ์

1 (mciRecordOverWrite) บันทึกข้อมูลทับข้อมูลเดิมของดีไวซ์

คุณสมบัติ Shareable

รายงานหรือกำหนดให้ดีไวซ์ MCI สามารถที่จะถูกใช้งานได้โดยหลาย ๆ แอพพลิเคชันในเวลาเดียวกัน (share) หรือไม่

```
[form.]MMControl.Shareable [= {True/False}]
```

คุณสมบัติ Silent

รายงานหรือกำหนดสถานะการเล่นกลับเสียงของดีไวซ์ MCI

```
[form.]MMControl.Silent [= {True/False}]
```

คุณสมบัติ Start

รายงานตำแหน่งเริ่มต้นของดีไวซ์ MCI ปัจจุบันที่ถูกเปิด โดยที่หน่วยของตำแหน่งปัจจุบันจะเป็นไปตามที่ถูกกำหนดโดยคุณสมบัติ TimeFormat

```
[form.]MMControl.Start
```

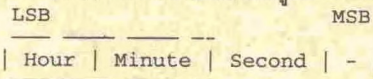
คุณสมบัติ TimeFormat

รายงานหรือกำหนด หน่วยสำหรับการรายงานตำแหน่งปัจจุบันของดีไวซ์ MCI

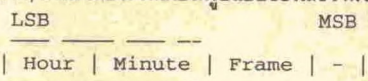
```
[form.]MMControl.TimeFormat [= format&]
```

0 (mciFormatMilliseconds) มิลลิวินาที (1/1000 วินาที)

1 (mciFormatHms) ชั่วโมง, นาที, วินาที โดยจะจัดเก็บในรูปแบบของเลขจำนวนเต็ม long (4 ไบต์) โดยที่แต่ละไบต์มีความหมายดังนี้

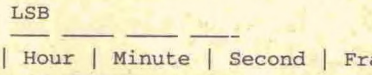


2 (mciFormatMsf) ชั่วโมง, วินาที, เฟรม โดยจะจัดเก็บในรูปแบบของเลขจำนวนเต็ม long (4 ไบต์) โดยที่แต่ละไบต์มีความหมายดังนี้

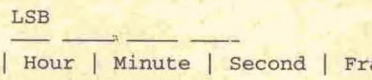


3 (mciFormatFrames) เฟรม

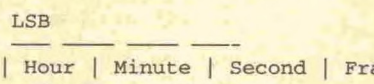
4 (mciFormatSmpte24) 24-frame SMPTE โดยจะจัดเก็บในรูปแบบของเลขจำนวนเต็ม long (4 ไบต์) โดยที่แต่ละไบต์มีความหมายดังนี้



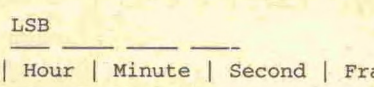
5 (mciFormatSmpte25) 25-frame SMPTE โดยจะจัดเก็บในรูปแบบของเลขจำนวนเต็ม long (4 ไบต์) โดยที่แต่ละไบต์มีความหมายดังนี้



6 (mciFormatSmpte30) 30-frame SMPTE โดยจะจัดเก็บในรูปแบบของเลขจำนวนเต็ม long (4 ไบต์) โดยที่แต่ละไบต์มีความหมายดังนี้



7 (mciFormatSmpte30Drop) 30-drop-frame SMPTE โดยจะจัดเก็บในรูปแบบของเลขจำนวนเต็ม long (4 ไบต์) โดยที่แต่ละไบต์มีความหมายดังนี้



→

8 (mciFormatBytes)	ไบต์
9 (mciFormatSamples)	จำนวนการสุ่มตัวอย่าง
10 (mciFormatTmsf)	แทร็ก, นาที, วินาที, เฟรม โดยจะจัดเก็บในรูปแบบของเลขจำนวนเต็ม long (4 ไบต์) โดยที่แต่ละไบต์มีความหมายดังนี้

```
LSB                                  MSB
___ ____ ____ ____
| Track | Minute | Second | Frame |
___ ____ ____ ____
```

คุณสมบัติ Track

รายงานหรือกำหนดตำแหน่งแทร็ก ที่ต้องการอ่านขนาดและตำแหน่งของแทร็กเท่านั้น ไม่เกี่ยวข้องกับแทร็กปัจจุบันของดีไวซ์

```
[form.]MMControl.Track [= track&]
```

คุณสมบัติ TrackLength

รายงานขนาดความยาวของแทร็ก โดยที่ผู้อ่านสามารถกำหนดตำแหน่งได้จากคุณสมบัติ Track

```
[form.]MMControl.TrackLength
```

คุณสมบัติ TrackPosition

รายงานตำแหน่งเริ่มต้นของแทร็ก โดยที่ผู้อ่านสามารถกำหนดตำแหน่งได้จากคุณสมบัติ Track

```
[form.]MMControl.TrackPosition
```

คุณสมบัติ Tracks

รายงานจำนวนแทร็กของดีไวซ์ที่ถูกเปิด

```
[form.]MMControl.Tracks
```

คุณสมบัติ UpdateInterval

รายงานเวลาในหน่วยของมิลลิวินาที สำหรับการเรียกรูทีนเหตุการณ์ StatusUpdate

```
[form.]MMControl.UpdateInterval [= milliseconds%]
```

โดยถ้าหากค่าของ *milliseconds%* เท่ากับ 0 ก็แสดงว่าจะไม่มีการเรียกรูทีนเหตุการณ์ StatusUpdate

คุณสมบัติ UsesWindows

รายงานถึงความต้องการใช้หน้าต่างใหม่ในการแสดงผลของดีไวซ์ที่ถูกเปิดหรือไม่

```
[form.]MMControl.UsesWindows
```

คุณสมบัติ Visible

รายงานหรือกำหนดสถานะการแสดงผลของชุดปุ่มคำสั่ง MCI คอนโทรล MCI

```
[form.]MMControl.Visible [= {True/False}]
```

คุณสมบัติ Wait

รายงานหรือกำหนดสถานะการอให้คำสั่ง MCI ทำงานจนเสร็จสมบูรณ์ก่อนที่จะคืนกลับการควบคุมมายังแอพพลิเคชันที่เรียกใช้คอนโทรล MCI หรือไม่

```
[form.]MMControl.Wait [= {True/False}]
```

เหตุการณ์ *Button*Click

เหตุการณ์นี้เกิดขึ้นเมื่อผู้ใช้มีการคลิ๊กที่ปุ่มใดปุ่มหนึ่งของคอนโทรล MCI

```
Private Sub MMControl_ButtonClick (Cancel
As Integer)
```

สำหรับรายชื่อของแต่ละปุ่มที่สนับสนุนมีดังนี้ Back, Eject, Next, Pause, Play, Prev, Record, Step และ Stop ทั้งนี้ก็ขึ้นอยู่กับชนิดของมัลติมีเดียที่คอนโทรล MCI กำลังติดต่อด้วยเช่นกัน

เหตุการณ์ *Button*Completed

เหตุการณ์นี้เกิดขึ้นเมื่อคำสั่ง MCI ที่ถูกปฏิบัติโดยที่ผู้ใช้มีการคลิ๊กที่ปุ่มใดปุ่มหนึ่งของคอนโทรล MCI สำเร็จสมบูรณ์

```
Private Sub MMControl_ButtonCompleted
(Errorcode As Long)
```

เหตุการณ์ *Button*GotFocus

เหตุการณ์นี้เกิดขึ้นเมื่อปุ่มใดปุ่มหนึ่งของคอนโทรล MCI ได้รับโฟกัส หรือหมายถึงมีปุ่มคำสั่งหนึ่ง ๆ เป็นปุ่ม default นั่นเอง

```
Private Sub MMControl_ButtonGotFocus ()
```

เหตุการณ์ *Button*LostFocus

เหตุการณ์นี้เกิดขึ้นเมื่อปุ่มใดปุ่มหนึ่งของคอนโทรล MCI สูญเสียโฟกัส หรือหมายถึงมีปุ่มคำสั่งหนึ่ง ๆ เป็นปุ่มเปลี่ยนสถานะจาก default เป็น non-default นั่นเอง

```
Private Sub MMControl_ButtonLostFocus ()
```

เหตุการณ์ Done

เหตุการณ์นี้เกิดขึ้นเมื่อคำสั่ง MCI ใด ๆ ที่มีการกำหนดให้คุณสมบัติ Notify มีค่าเท่ากับ True ทำงานเสร็จสมบูรณ์ ซึ่งเหตุการณ์ที่ทำให้ VB/Win สามารถทราบถึงสถานะการทำงานของแต่ละคำสั่ง

```
Private Sub MMControl_Done (NotifyCode As Long)
```

โดยที่ *NotifyCode* สามารถมีค่าได้ดังต่อไปนี้

1 (mciNotifySuccessful) คำสั่งทำงานเสร็จเรียบร้อย

2 (mciNotifySuperseded) คำสั่งถูกแทนที่หรือควบคุมโดยคำสั่งอื่น ๆ

3 (mciNotifyAborted) คำสั่งถูกยกเลิกโดยผู้ใช้

8 (mciNotifyFailure) คำสั่งทำงานไม่สำเร็จ

เหตุการณ์ StatusUpdate

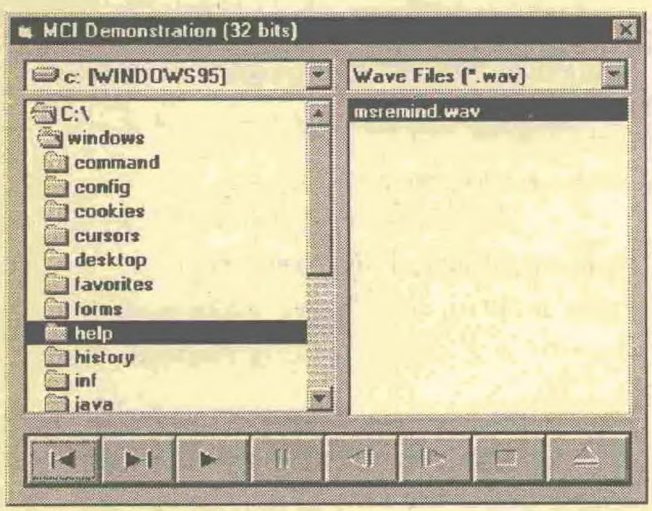
เหตุการณ์นี้เกิดขึ้นเมื่อต้องการให้คอนโทรล MCI ทำการอัพเดตอัตโนมัติตามช่วงเวลาที่กำหนดโดยคุณสมบัติ UpdateInterval

```
Private Sub MMControl_StatusUpdate ()
```

โปรแกรมตัวอย่าง MCIDEMO

จากข้างต้นทั้งหมดผู้เขียนก็ได้อธิบายถึงคุณสมบัติและเหตุการณ์ต่าง ๆ ที่คอนโทรล MCI สนับสนุน ดังนั้นในส่วนสุดท้ายของบทความตอนนี้ ผู้เขียนจึงได้สร้างโปรแกรมตัวอย่าง MCIDEMO เพื่อให้ผู้อ่านสามารถเข้าใจถึงวิธีการใช้งานคอนโทรล MCI32.OCX มากขึ้น โดยที่โปรแกรมตัวอย่าง MCIDEMO จะเป็นโปรแกรมสั้น ๆ ที่ประกอบด้วยฟอร์มเพียง 1 ฟอร์มเท่านั้น โดยในรูปที่ 4 แสดงถึงตัวอย่างการเล่นกลับไฟล์ .wav ที่ชื่อ msremind.wav โดยที่โปรแกรมตัวอย่างนี้ผู้เขียนเจาะจงเฉพาะให้สนับสนุนดีไวซ์ MCI ดังต่อไปนี้ WaveAudio, CDAudio, MMMovie, Sequencer และ AVIVideo เท่านั้น

ซึ่งการทำงานของโปรแกรมก็ง่าย ๆ โดยเริ่มจากเมื่อผู้อ่านรันโปรแกรม รูทีนเหตุการณ์ Form_Load ก็จะถูกเรียกซึ่งในรูทีนนี้ก็จะมีการเขียนโค้ดเพื่อย้ายฟอร์มมาแสดงผลที่กลางจอภาพ และเพิ่มรายการมัลติมีเดียตัวอย่าง ให้กับคอนโทรล Combo1 ดังนี้



รูปที่ 4 แสดงถึงตัวอย่างการเล่นกลับไฟล์ .wav

```
    ' Show form at middle of screen.
    Move (Screen.Width - Width) / 2,
(Screen.Height - Height) / 2

    ' add multimedia list to combo box.
    Combo1.AddItem "Wave Files (*.wav)"
    Combo1.AddItem "MIDI Files (*.mid)"
    Combo1.AddItem "MMM Files (*.mmm)"
    Combo1.AddItem "AVI Files (*.avi)"
    Combo1.AddItem "CD Audio"
    Combo1.ListIndex = 0
```

จากนั้นผู้อ่านก็สามารถเลือกชนิดของดีไวซ์ที่ต้องการได้จาก Combo1 โดยเมื่อมีการเปลี่ยนชนิดของดีไวซ์ก็จะทำให้เกิดเหตุการณ์ Combo1_Click ทันที ซึ่งภายในรูทีนนี้ก็จะมีการเขียนโค้ดเพื่อปิดดีไวซ์ที่ถูกเปิดอยู่แล้วก่อนหน้านี้ (ถ้ามี) โดยการตรวจสอบจากคุณสมบัติ Mode ดังนี้

```
Dim i%, p$

' If the device is open, close it.
If Not MMControl1.Mode = mciModeNotOpen Then
    MMControl1.Command = "Close"
End If
MMControl1.filename = ""
```

ต่อมาก็กำหนดชนิดของดีไวซ์ใหม่ที่ถูกเลือกให้กับคุณสมบัติ DeviceType และถ้าหากดีไวซ์ที่ถูกเลือกไม่ใช่ CDAudio ก็แสดงว่าต้องกำหนดชื่อไฟล์ให้กับคุณสมบัติ Filename ซึ่งจะถูกจัดการในรูทีน File1_Click

```
    Select Case Combo1.ListIndex
    Case 0: MMControl1.DeviceType = "WaveAudio"
    Case 1: MMControl1.DeviceType = "Sequencer"
    Case 2: MMControl1.DeviceType = "MMMovie"
    Case 3: MMControl1.DeviceType = "AVIVideo"
    Case 4: MMControl1.DeviceType = "CDAudio"
    End Select

    i% = InStr(Combo1.Text, "(")
    If i% Then
        File1.Pattern = Mid$(Combo1.Text, i% +
1, Len(Combo1.Text) - i% - 1)
        File1.Refresh
    Else
        MMControl1.TimeFormat = mciFormatTmsf
        MMControl1.Command = "Open"
    End If
    MMControl1.Refresh
```

แต่เพื่อความเข้าใจก็ขอให้ผู้อ่านทดลองรันโปรแกรมตัวอย่างเอาเอง เพราะจะง่ายที่การเข้าใจบทความตอนนี้ และถ้าหากผู้อ่านมีเวลาก็ทดลองแก้ไขซอร์สโค้ดโดยการเพิ่มชนิดของดีไวซ์ให้มากขึ้นก็จะทราบถึงวิธีการใช้งานคอนโทรล MCI ได้ดีขึ้นมากทีเดียว ก่อนจบบทความก็ขอแจ้งให้ผู้อ่านที่ต้องการติดต่อกับผู้เขียนก็ขอให้เขียนจดหมายมาที่หมายเลขตู้ป.ณ.ใหม่ดังนี้ครับ ตู้ป.ณ. 95 ปทฝ.ดินแดง กรุงเทพฯ 10407 สุดท้ายก็ขอให้ผู้อ่านติดตามบทความในตอนต่อ ๆ ไปครับ พบกันใหม่ในตอนต่อไปสวัสดีครับ

ลิสติง 1 ไฟล์โปรเจ็กต์ MCIDEMO.VBP

```
Form=MCIDemo.Frm
Object={C1A8AF28-1257-101B-8FB0-0020AF039CA3}#1.0#0; MCI32.OCX
Reference=*\G{BEF6E001-A874-101A-8BBA-
00AA00300CAB}#2.0#0#C:\WINDOWS\SYSTEM\OLEPRO32.DLL#Standard OLE Types
ProjWinSize=100,0,248,215
ProjWinShow=2
IconForm="Form1"
Title="MCIDemo"
ExeName="WTDEMO.EXE"
Path=".."
Name="WTDEMO"
HelpContextID="0"
StartMode=0
VersionCompatible32="0"
VersionCompatible="0"
MajorVer=1
MinorVer=0
RevisionVer=0
AutoIncrementVer=0
ServerSupportFiles=0
VersionComments="Distributed by Microcomputer Magazine _
                 Published by SE-ED Co.ltd."
VersionCompanyName="SE-ED Co.,Ltd."
VersionFileDescription="Multimedia Control Interface"
VersionLegalCopyright="Mr.Sutthisak Phongthanapanich"
```

ลิสติง 2 ไฟล์ฟอร์ม MCIDEMO.FRM

```
VERSION 4.00
Begin VB.Form Form1
   BorderStyle     =   3  'Fixed Dialog
   Caption         =   "MCI Demonstration (32 bits)"
   ClientHeight    =   4230
   ClientLeft      =   1620
   ClientTop       =   1515
   ClientWidth     =   5880
   BeginProperty Font
      name            =   "MS Sans Serif"
      charset         =   222
      weight          =   700
      size            =   8.25
      underline       =   0   'False
      italic          =   0   'False
      strikethrough   =   0   'False
   EndProperty
   ForeColor       =   &H80000008&
   Height          =   4635
   Left            =   1560
   LinkTopic       =   "Form1"
   LockControls    =   -1  'True
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   4230
   ScaleWidth      =   5880
   ShowInTaskbar   =   0   'False
   Top             =   1170
   Width           =   6000
   Begin VB.ComboBox Combo1
      Height          =   315
      Left            =   3120
      Style           =   2  'Dropdown List
      TabIndex        =   4
      Top             =   120
      Width           =   2655
   End
   Begin VB.FileListBox File1
      Height          =   2985
```

→

ลิสติง 2 (ต่อ)

```
      Left            =   3120
      TabIndex        =   3
      Top             =   480
      Width           =   2655
   End
   Begin VB.DirListBox Dir1
      Height          =   2955
      Left            =   120
      TabIndex        =   2
      Top             =   480
      Width           =   2895
   End
   Begin VB.DriveListBox Drive1
      Height          =   315
      Left            =   120
      TabIndex        =   1
      Top             =   120
      Width           =   2895
   End
   Begin MCI.MMControl MMControl1
      Height          =   495
      Left            =   120
      TabIndex        =   0
      Top             =   3600
      Width           =   5655
      _Version        =   65536
      RecordVisible   =   0   'False
      _ExtentX        =   9975
      _ExtentY        =   873
      _StockProps     =   32
      BorderStyle     =   1
   End
End

Attribute VB_Name = "Form1"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit
```

→

ลิสติง 2 (ต่อ)

```
' Project name   : MCIDEMO.VBP
' Program name   : Multimedia Control Interface.
' VB/Win Module  : MCIDEMO.FRM
' DLL filename   : MMSYSTEM.DLL
' Copyright (C)1997 by SUTTHISAK PHONGTHANAPANICH
' Distributed by: SE-ED Co.,Ltd.
'

'
' Windows API functions & constants.
'

Private Sub Combo1_Click()
    Dim i%, p$

    ' If the device is open, close it.
    If Not MMControl1.Mode = mciModeNotOpen Then
        MMControl1.Command = "Close"
    End If
    MMControl1.filename = ""

    Select Case Combo1.ListIndex
    Case 0: MMControl1.DeviceType = "WaveAudio"
    Case 1: MMControl1.DeviceType = "Sequencer"
    Case 2: MMControl1.DeviceType = "MMMovie"
    Case 3: MMControl1.DeviceType = "AVIVideo"
    Case 4: MMControl1.DeviceType = "CDAudio"
    End Select

    i% = InStr(Combo1.Text, "(")
    If i% Then
        File1.Pattern = Mid$(Combo1.Text, i% + 1, _
                        Len(Combo1.Text) - i% - 1)
        File1.Refresh
    Else
        MMControl1.TimeFormat = mciFormatTmsf
        MMControl1.Command = "Open"
    End If
    MMControl1.Refresh
End Sub

Private Sub Dir1_Change()
    File1.Path = Dir1.Path
End Sub

Private Sub Drive1_Change()
    Dir1.Path = Drive1.Drive
End Sub

Private Sub File1_Click()
    If File1.ListIndex >= 0 Then
        On Error Resume Next

        Dim mciFilename$

        If Right$(Dir1.Path, 1) <> "\" Then
            mciFilename$ = Dir1.Path & "\"
        Else
            mciFilename$ = Dir1.Path
        End If
        MMControl1.filename = mciFilename$ & _
                        File1.List(File1.ListIndex)

        ' If the device is open, close it.
      If Not MMControl1.Mode = mciModeNotOpen Then
            MMControl1.Command = "Close"
        End If
        MMControl1.Command = "Open"
        MMControl1.From = 0
    End If
End Sub
```

→

ลิสติง 2 (ต่อ)

```
Private Sub Form_Load()
    ' Show form at middle of screen.
    Move (Screen.Width - Width) / 2, (Screen.Height
- Height) / 2

    ' add multimedia list to combo box.
    Combo1.AddItem "Wave Files (*.wav)"
    Combo1.AddItem "MIDI Files (*.mid)"
    Combo1.AddItem "MMM Files (*.mmm)"
    Combo1.AddItem "AVI Files (*.avi)"
    Combo1.AddItem "CD Audio"
    Combo1.ListIndex = 0
End Sub

Private Sub Form_Unload(Cancel As Integer)
    On Error Resume Next

    MMControl1.Command = "Close"
    End
End Sub

Private Sub MMControl1_PlayClick(Cancel As Integer)
    If MMControl1.DeviceType <> "CDAudio" Then
        MMControl1.From = 0
    End If
End Sub
```

■

เทคนิคประจำคอลัมน์

ถาม ไฟล์ Autorun.inf มีหน้าที่อะไร?

ตอบ ไมโครซอฟท์ได้เพิ่มความสามารถใหม่ให้กับวินโดวส์ 95 ที่เรียกกันว่า AutoPlay ซึ่งเป็นดีไวซ์ของวินโดวส์ 95 ที่ทำหน้าที่ตรวจสอบ CD-ROM เมื่อเราทำการโหลดแผ่น CD ให้เครื่องคอมพิวเตอร์ โดยที่วินโดวส์ 95จะตรวจสอบไฟล์ที่ชื่อ Autorun.inf จากนั้นวินโดวส์ 95 ก็จะทำงานตามคำสั่งที่ปรากฏในไฟล์ดังกล่าว เช่นสมมติว่ามีคำสั่งดังนี้

```
[autorun]
OPEN=INSTALL.EXE
ICON=SEED.ICO
```

วินโดวส์ 95 ก็จะทำการโหลดไฟล์ INSTALL.EXE และกำหนดให้ใช้ไอคอนจากไฟล์ SEED.ICO ในการแสดงผลไฟล์ดังกล่าว ส่วนสาเหตุที่วินโดวส์ 95 สนับสนุนดีไวซ์ AutoPlay ก็เพื่อให้ผู้ใช้ที่เป็นมือใหม่จริงสามารถทำการติดตั้งแอพพลิเคชันจากแผ่น CD ได้ง่ายขึ้น โดยไม่จำเป็นต้องเรียกไฟล์ SETUP.EXE ให้ยุ่งยาก